Evaluating HyTime: An Examination and Implementation Experience

John F. Buford

Distributed Multimedia Systems Laboratory
Department of Computer Science
University of Massachusetts Lowell
One University Avenue
Lowell, MA 01854USA
URL: http://dmsl.cs.uml.edu
E-mail: buford@cs.uml.edu

ABSTRACT

HyTime defines an extensive meta-language for hypermedia documents, including general representations for links and anchors, a framework for positioning and projecting arbitrary objects in time and space, and a structured document query language. We propose a set of criteria for evaluating the HyTime model. We then review the model with respect to these criteria and describe our implementation experience. Our review indicates both the benefits and limitations of HyTime. These results are relevant to systems and applications designers who are considering HyTime, and also to possible future revisions of the standard.

Keywords: HyTime, hypermedia models, hypermedia standards

INTRODUCTION

Since 1991 we have developed various hypermedia systems and applications which are based on the HyTime ISO standard [17]. This work includes:

1. A HyTime engine based on an OODBMS [1]
2. Examination of automatic HyTime application generation[5]
3. Examination of different approaches for integrating multimedia scripting languages with HyTime[3]
4. Potential relationships between HyTime and HTML[25][27]
5. An implementation and evaluation of the HyQ query language[6]
6. An architecture for a distributed delivery model [7][8]
7. A graphical representation for HyTime document design [26]
8. A proposal for an open hyperdocument architecture based on HyTime [7]

In this paper we present an evaluation of HyTime based on this research experience and with some consideration for the requirements for hypermedia document architecture as developed by various hypermedia researchers.

The HyTime specification was developed in the context of hypertext practice in the late 1980s. Additionally, musical time models, originally developed for the SMDL (Standard Musical Description Language) standard, were generalized in HyTime's scheduling and projection modules[24]. Further, some facilities in HyTime are improvements to SGML (Standard Generalized Markup Language). Since this time, significant developments have occurred in the design of distributed hypermedia systems, multimedia media formats, multimedia scripting languages and authoring paradigms. Also, new distributed object frameworks are providing an environment for creating hyperapplications. These developments provide a new context in which to consider the design and use of HyTime.

Prior to its standardization, little of HyTime was validated by implementation. Subsequent to its standardization, only a few implementations of subsets of HyTime exist, in part because of the complexity of the specification. Complete validation would require significant experience with:

Interchange between different hypermedia systems

Multimedia DTD (document type definition) design

Creation of large heterogeneous hypermedia document collections

Nevertheless a useful assessment can be made based on the criteria that we present in a later section. This assessment is valuable for designers of hypermedia systems, hypermedia documents, and other hypermedia standards, as well as for those interested in HyTime.

Hypertext '96, Washington DC USA
© 1996 ACM 0-89791-778-2/96/03..$3.50

In this paper we present an analysis of HyTime as a meta-language for hypermedia documents. We define our evaluation criteria, and then examine HyTime according to each of these criteria, noting both positive and negative aspects. Since HyTime has been described in several places [9][24], we omit a HyTime tutorial. There is an ongoing activity to develop conventions for the use of HyTime by the user group known as CaPH. We do not address this work in this paper. A number of HyTime facilities are extensions or improvements of SGML features. Since our interest is in hypermedia semantics, we do not address this particular area in this paper. The next section provides an overview evaluation, drawing upon the context and basic philosophy of HyTime's development. Section three present a set of criteria for evaluating HyTime. Subsequent sections discuss HyTime with respect to these criteria. The last section summarizes the paper.

SOME GENERAL OBSERVATIONS

When HyTime was developed, the practice of hypertext could be characterized as stand-alone systems which used different models for document structure, anchors, links, navigation, and other key features. A basic problem was the ability to interchange hypertext documents between systems with different models of hypertext. HyTime was in part conceived as a solution to the problem of interchange of content between different hypertext systems. For interchange purposes, a given hypertext system could define a HyTime DTD and then export content as a document instance for this DTD. A system receiving this document instance, which in this example would have defined its own HyTime DTD, would attempt to translate the elements and attributes of the imported document to corresponding elements and attributes of the native system. In so far as each DTD is able to use HyTime facilities for representing these elements and attributes, the translation would be simplified and the amount of custom software and manual intervention would be minimized.

HyTime was developed by the same committee that created the SGML standard, and the primary target of HyTime is the SGML community. In particular, HyTime itself is defined as an SGML meta-DTD, and use of HyTime requires use of SGML. Now with the rapid evolution of the world-wide web (WWW), there is also the prospect that portions of the much larger HTML community will migrate to SGML. HyTime is a natural evolutionary direction for an SGML web. We discuss this scenario in previous work [6] and briefly evaluate the relationship between HyTime and WWW later in this paper.

HyTime is a meta-language for creating new hypermedia document types. An application designer identifies the structural elements that are suitable for the target document type. When constructing the application DTD, the designer uses the corresponding HyTime architectural forms which fit the semantics of the corresponding elements and attributes. Application structure which is not representable with HyTime facilities can be represented by defining application specific elements and attributes.

The meta definition is a syntactic mechanism and does not increase the semantic capability of HyTime. The meta-DTD definition of HyTime has some benefits when compared to the alternative of defining a single standard hypermedia DTD:

1. Backward compatibility with existing SGML applications

The meta definition makes it easier to incorporate facilities of HyTime into existing SGML DTDs without modifying the names of existing elements and attributes

2. Multiple HyTime-based DTDs

Many different application DTDs can be defined as applications of HyTime. Similarly, the instances of many different DTDs can be included in the same HyTime hypermedia document set.

3. The name space for element generic identifiers (GI) is more flexible

The HyTime meta-DTD doesn't define any GIs, and more than one application GI can use a HyTime ETF (Element Type Form). Additionally, an application can define its own non-HyTime GIs, and has complete control over attribute names.

The meta definition, however, complicates the use of HyTime since it leads to an additional level of indirection. That is, the designer of a tag set must conceptually work at one additional level, the meta-DTD level, when defining elements and attributes for the application DTD.

Previously [7] we have argued that HyTime, as a standard for hypermedia document models, offers the possibility of increased generality in the design of hypermedia delivery systems. This is shown in Figure 1.

Since many hypermedia applications share structural and composition requirements--in particular time, space, and hyperlinks--a standard description of these components of applications means that less custom code is needed to deliver the application. At the same time, HyTime is almost entirely silent about presentation and interaction, and so is an incomplete way of describing most existing hypermedia and multimedia documents. Consequently it is impossible to deliver any hypermedia application by providing only a HyTime document instance and DTD to a HyTime engine. Unlike SGML applications which operate with a document instance, DTD, and style sheet, typical

impossible to deliver any hypermedia application by providing only a HyTime document instance and DTD to a HyTime engine. Unlike SGML applications which operate with a document instance, DTD, and style sheet, typical HyTime applications appear to require custom software beyond the HyTime engine [7]. This custom software is not just for the media specific presentation. Custom software is required for all time, space, and interaction semantics for presentation. This means that each time a new HyTime DTD is created for a new application, some custom presentation software must also be developed. We discuss a way to mitigate this later in the implementation section. In previous work [7] we have investigated the feasibility of automatically generating some or all of the application presentation software from the application DTD. By adopting certain conventions, a significant portion of the presentation software could be automatically generated, but it is not a complete solution.

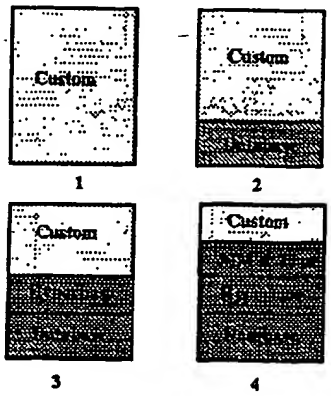

Figure 1. Progression of generality in hypertext engine design: 1) completely custom architecture, 2) storage of hypermedia information in a backend database (first done in mid-1980s), 3) exporting a link database on top of the database, 4) supporting generic multimedia and temporal modeling constructs. Most systems today are at stage three. A HyTime engine is a stage four system

CRITERIA

Standardization and the Space of Hypermedia Functionality

Any attempt to standardize a hypermedia document architecture must please a diverse set of applications. Additionally, international standards have a projected lifetime of five to ten years, and it is a challenging task for the designers of a standard to specify a framework which will remain adequate in the face of an evolving field. Further, standards frequently define many options or profiles which permit a flexible range of uses but at the same time may appear as unnecessary minutia to the observer. Finally, compatibility with other standards is also an important consideration.

The question of what should be standardized and what should be left for applications to decide is a crucial one. Figure 2 shows a high level view comparing HyTime facilities with the space of hypermedia functionality. In this space HyTime focuses on representing structure of both the hypergraph and the nodes of the hypergraph. Delivery issues including interaction and presentation facilities, integration with computation engines and other applications, and authoring paradigms are outside the scope of HyTime.

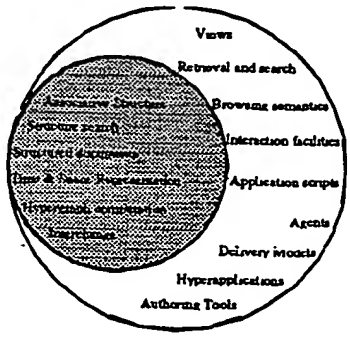

Figure 2. HyTime facilities as a subset of hypermedia systems functionality

Criteria

Each of the following sections examines HyTime according to one of the following perspectives:

1. Representational completeness

The most basic question is whether HyTime is representationally complete for the functionality it standardizes. This question can be addressed by developing a statement of requirements and comparing HyTime with this statement, and by comparing HyTime with other formal models of hypermedia that have been developed. In this paper we take this latter approach. A related question is whether a HyTime system is operationally open and extensible.

2. Technical correctness and completeness

Given the scope of representation, the next question is whether HyTime modules and architectural forms (AFs) correctly and completely provide the intended scope. This question can be partially addressed by interpreting the specification, and by using HyTime to encode specific hypermedia document structures.

3. Relationship to the WWW and other more recent developments in the practice of hypertext

Another measure of HyTime is its flexibility for use in contexts which have emerged subsequent to the

specification of the standard. These contexts might include:

Virtual reality, e.g., the use of HyTime to represent hyperlinking in/between virtual worlds

Integration with content-based retrieval methods, e.g., to create anchors to non-text media

Hyperapplications, e.g., the relevance of HyTime to representing hyperlinks between applications

4. Implementation experience

Implementation experience with HyTime can be used for determining the efficiency of systems which process HyTime for delivery and interchange, and for evaluating the complexity of the HyTime processing architecture versus the resulting functionality.

REPRESENTATIONAL COMPLETENESS

Comparison to the Dexter Hypertext Model

The Dexter hypertext model is a formal model developed in the late 1980s as a generalization of the leading contemporary hypertext systems [13]. The model divides hypertext delivery systems into three layers, with most of the emphasis on the storage layer. A key insight of Dexter is the identification of anchors and links as distinct entities in the hypergraph. In Dexter the anchor encapsulates the content-specific address of the link endpoint, isolating the link from the details of how the content is organized. Subsequent work has extended the Dexter model to permit dangling links and the ability to define anchors for composite nodes.

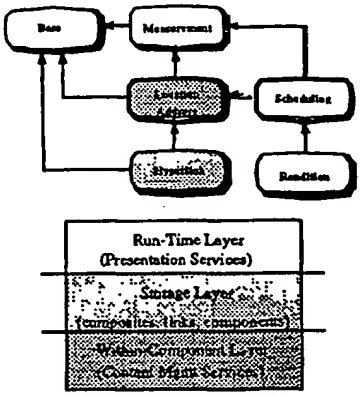

Figure 3. Overlap between the six HyTime modules and the three layers of the Dexter model

Figure 3 shows the overlap between the six HyTime modules and the three layers of Dexter. The within-component layer corresponds to the HyTime location address module, and the storage layer corresponds to the hyperlinks module. Like Dexter anchors, HyTime location address forms isolate links from the details of content-specific addressing. HyTime, however, provides many different addressing techniques that can be used to form an anchor (Table 1). These forms can be directly applied to any text content. An application can also define its own content specific notations and create HyTime anchors which use these notations. A set of different addresses can be aggregated (aggloc) in various ways, and a set of addresses can define the span of an anchor (spanloc).

HyTime introduces the notion of a location ladder which generalizes the concept of an anchor. Any location reference can refer to an already existing location, and an arbitrary series of location references can be chained together. This allows a newer anchor, perhaps created in a different document owned by another user, to build upon existing anchors. HyTime also permits anchors to be formed dynamically using the HyQ query language described in the next section.

Table 1. Summary of HyTime addressing forms

Address Attribute	Brief Description
Name space	Position based on unique name associated with document node
Data	Position based on treating content as ordered units of quanta
Node	Position in document structure, including locating by position in: 1) document tree, 2) one or more paths in a tree; 3) position in a list of nodes of a document; 4) relative position with respect to some other node
Property	Position based on property attributes associated with nodes
Bibliographic	A bibliographic reference to off-line material

HyTime generalizes Dexter's link and anchor model, adding many mechanisms for addressing content as well as permitting application define their own content-addressing notations. HyTime permits dangling links, and allows anchors to refer to composite nodes. HyTime anchors can also reference objects that are positioned in time or space. HyTime links and anchors can be placed in separate documents from which the addressed content is stored.

HyTime doesn't provide a standard mechanism for applications to associate presentation attributes or

Time Model

Time is a fundamental compositional notion in multimedia and hypermedia documents. Although HyTime has a scheduling module, there is no explicit time or space axes, but simply arbitrary dimensions which the application is free to interpret as it wishes. For example, one might place a series of images in a HyTime axis named "t" and associate units of seconds with this axis, but HyTime doesn't prescribe whether these images should be shown in time order to the user, or whether they just happen to have an intrinsic chronological relation, such as the order in which they were painted by an artist. The interpretation of the axis in time or space is left to the application.

Erfle [10] has evaluated HyTime's coordinate axes and position mechanisms with respect to the requirements of temporal composition, and concludes that for fifteen issues of interest, HyTime is compositionally sufficient. Recent work [15] in translating the Amsterdam Hypermedia Model to HyTime shows that HyTime is adequate for representing the parallel time channels and synchronization edges used in AHM.

One important area is the ability to represent synchronization relationships. In multimedia documents, the specification of a synchronization relationship between two or more media allows the presentation system to properly coordinate synchronization recovery in the event of unexpected delay in one or more of the media channels. The HyTime standard states (clause 7.4):

"It can be useful to specify all or part of a dimension in terms of components of another dimension. For some applications, this technique could be used to represent alignment and synchronization relationships."

The dimension of one event in a HyTime schedule can be defined relative to another event. If, during presentation delivery, a presentation delay occurred for the referenced event, then a delivery system might infer that the relative event should also be delayed or resynchronized. However, there is no way within HyTime for an application to specify that such alignment relationships should be interpreted as synchronization points.

In the case of temporal structure, HyTime provides facilities which can be used to represent both temporal composition and temporal synchronization, but the temporal semantics are left to the application to determine.

Comparison to Halasz' Third Generation System

Halasz [14] described seven areas which third generation hypermedia systems must address. These issues, and the relationship of HyTime to them, are shown in Table 2. HyTime appears to be suitable for interchange between second-generation hypermedia systems.

Issue	Description	HyTime Support
Integration of Search and Query Functionality	Integration of information retrieval and DBMS facilities; pattern matching languages for hypergraph manipulation	Partial, through the availability of HyQ, the match function, and application-defined query notations
Composite Node Types	In addition to content nodes, need container or collection nodes	Yes; each non-leaf node can be a composite (also in SGML)
Virtual Structures over Node Collections	Computationally-defined hypergraphs, analogous to database views	Partial, through use of the HyQ query language combined with linking and location addressing
Computation over Hypermedia Networks	Integrated computational engines available to the application	No; must be an external process; no specific features for integrating with external processes
Versioning of Nodes and Subgraphs	Maintaining change history at both node and graph level; version identification as an attribute in search and query	No; not considered to be within the scope of the standard
Support for Collaborative Work	Support for shared access to hypertext, and group protocols	No; not considered to be within the scope of the standard
Extensibility and Tailorability	Easier customization of hypermedia system by end user	Partial, through the definition of new DTDs

Table 2. Halasz's [14] seven issues for third-generation hypermedia systems

Hypermedia Structure, Semantics, and Presentation and Interaction

A fundamental tenet of SGML and HyTime is that logical structure of documents which have archival value should be separated from presentation specific attributes. SGML applications use a style sheet to define layout and format, and many different style sheets could be used with the same document instance.

Multimedia and hypermedia applications are only partially about logical content organization. They are also about creating an interactive audio-visual experience, and the artist who develops a multimedia presentation might consider the stylistic aspects intrinsic to the composition. As observed earlier, HyTime defines no mechanisms for integrating application scripts or for defining interaction or presentation attributes. It has limited support for associating browsing semantics with the hypergraph structure.

HyTime takes a conservative position as to the need to represent presentation and interaction semantics, and provides structuring mechanisms with little semantics. Consequently, HyTime is not a self-contained vehicle for interchanging hypermedia and multimedia information. While it provides a declarative notation which is desirable for archival and processing purposes, its limited semantics lead to increased application-specific processing, whether for presentation or for retrieval.

SELECTED DESIGN FEATURES AND TECHNICAL CORRECTNESS

Modularity and Incremental Use

The HyTime structuring language includes sixty-nine element type forms (ETF) and twenty-four attribute list forms (ALF). Many applications of HyTime need only a subset of these facilities. Through the use of HyTime support declarations, an application DTD can identify which modules and options are required for the application to be properly interpreted.

It is also possible for an SGML application to use individual HyTime architectural forms, but such an application won't be a conforming application.

A stated objective of HyTime's support declaration is to permit an application to require only the relevant subsets of HyTime. For example, the standard defines a continuum of five document declarations which range from minimum to fairly full use of the facilities of the six modules. We have developed a simple HyTime application called HMP [1] (Hypermedia Presentation) which uses only ten ETFs but has a support declaration which implies twenty-seven ETFs. Any interesting multimedia application will probably have a support declaration comparable to that of the Basic Schedule declaration--forty-eight AFs (architectural forms)--but might need significantly fewer.

Analysis of the dependencies between modules and architectural forms shows that many AFs can be used independently. The support declaration specification in HyTime is not sufficiently fine-grained to take advantage of the relative independence of AFs. A fine-grained approach, for example, would allow an application to list only the AFs that it needed. The advantage of a fine-grained approach would be that an engine implementing a small but useful subset of HyTime, such as the HMP DTD, could still be HyTime conforming.

HyQ

HyQ is a query language whose domain is one or more HyTime documents. A HyQ query returns node lists, where nodes are element structure within an SGML/HyTime document. HyQ can be used to form dynamic anchors, or could be used by an external application. HyQ is defined in an appendix of IS 10744. In previous work [6] we analyze HyQ through a series of twenty-five examples. We have also implemented a HyQ processing system and tested a number of these queries.

HyQ is a restricted functional language. It is a side-effect free language (except for the fairly restricted Assign operator). It has limited conditional execution constructs. It focuses mostly on manipulation by querying as opposed to manipulation by dynamic creation. It appears to permit recursive functions to be written, but because of the lack of conditional flow of control constructs, writing a termination condition for a recursive HyQ function appears tricky. It is clumsy at best to write HyQ queries which identify nodes which minimize or maximize some attribute. We note also that the inability to re-order node lists, for example by sorting them, seems to be a limitation. There are no arithmetic operators.

In previous work [6] we have analyzed the computational facilities of HyQ, in particular the ability to query the hypergraph network. We have developed HyQ examples for the following types of queries:

Obtaining the anchors of a link

Obtaining the links attached to a given anchor

Finding the anchors for a given anchor role for the hyperlink

Finding the depth one web from current document

Halasz' issue-position query [14]

Guided tour via keyword attribute or keyword in content of node

These queries use one or more of the node properties defined by HyTime as shown in Table 3. The interpretation

of these hyperlink properties in a multiple document context is not clearly defined by the standard. For example, it is not specified whether the linkedby property would identify all hyperlinks in all documents that use this anchor, or whether only a subset would be identified.

Name	Description in ISO 10744
hylink	ilink or clink element
anchors	objects linked by hylink
ANCHROLE	anchor roles of anchors of hylink
linkedby	hylinks that link an anchor
linkedto	other anchors of linkedby
LINKEDAS	role of anchors in hylink

Table 3. Link properties (uppercase/lowercase symbols as given in the standard)

It is possible to use HyQ to do sophisticated structure querying against the hyperlink web. Much of HyQ's power comes from its tight integration with HyTime's location addressing facilities summarized earlier. The content matching capability, based on the HyLex regular expression language, is weak compared to common information retrieval techniques. There is no specific support for integrating HyQ with retrieval functions that return ranked results.

There are ways to extend HyQ, as well as define an application-specific query language as an alternative to HyQ. This would limit the portability of the document.

A soon to be completed corrigendum of HyTime is expected to replace HyQ with the more powerful SPQL query language defined in the DSSSL standard.

Addressing Media Objects

The nature of multimedia documents is that much of the content of the document will be in binary files external to the HyTime document. HyTime's role in a multimedia document is to connect all the pieces of the presentation and represent the structural relationships between them. Important relationships may include spatial, temporal, composition, transformation, and hyperlinking. All media and non-HyTime files will be interpreted and presented by processes separate from the HyTime processing system. HyTime is neutral with respect to the formats used by non-HyTime files. It does provide architectural forms for supporting location addressing of objects in non-HyTime formats, and for representing hierarchical external entity structure when the media entity of interest is embedded in a container entity.

HyTime extends the SGML facilities for dealing with binary entities. HyTime defines two data content notation ALFs in the base module, sbento and any-dcn, which are data attributes for entities. Together they can be used to formally describe an external entity which contains nested entities. Common examples of this in multimedia data are frames in a video sequence, audio tracks in a movie file, and color tables in graphics files. We have created an example that uses sbento and some of the attributes of any-dcn to create an entity reference to a frame in an MPEG video sequence. MPEG, a recent ISO standard for compressed digital video and audio, composes its bitstream in a hierarchy, defined as follows:

A video sequence consists of one or more groups of pictures

A group of pictures consists of a set of pictures

A picture consists of a set of slices

A slice consists of a set of macroblocks

A macroblock consists of a set of blocks

A block is an 8x8 pixel array

The top two levels of the hierarchy permit random access to the video frame level. The lower levels are fundamental part of the encoding and decoding algorithm, but could also be used by the application to address sub-elements in a frame.

The sbento content addressing mechanism requires complete enumeration of the extents of all containers in the hierarchy. This may be impractical for lengthy continuous media sequences, or for encodings which are generated dynamically.

Rendition Module

In the rendition module, HyTime provides a framework for representing object modification and transformation. These facilities might be used to specify how media objects are to be rendered to the presentation space or the type of transition effect to use for displaying a media object. However, the rendition facilities are defined at an abstract level, and presentation semantics are left to the application to interpret. Similarly, location addressing of external media formats depends on formal notations being specified for those formats. These would be integrated with a HyTime document by adding an SGML notation declaration.

Activity Tracking

Selected portions of document structure can be monitored by the HyTime engine for specific types of user access, including the operations of creation, deletion,

modification, access, link, and unlink. The engine will report each type of access to the application. This innovative facility has many potential applications.

RELATIONSHIP TO OTHER RECENT DEVELOPMENTS

HTML and WWW

The ultimate validation of a standard is when a significant community of users adopts the standard. HyTime is a recent standard, and has had virtually no commercial impact as yet. Yet during the same period a much more limited hypertext model, the HyperText Markup Language, has grown explosively in use. HTML 2.0 is now defined as an SGML DTD.

HTML linking is a semantic subset of HyTime's location addressing and hyperlink modules. There is no comparable facility in HyTime for HTML forms. HyTime facilities for time-space positioning, projection, rendition, location addressing, etc., have no counterpart in HTML. Rutledge et al. [25] have shown that HTML can be defined as a HyTime DTD [27].

The need for archival content delivered via WWW may lead to web viewers providing general SGML support. Activities such ACM's electronic publishing proposal and the Text Encoding Initiative (TEI) are examples of electronic publishing applications for which HTML is insufficient. The W3O developer libraries are being architected to support client-side SGML parsing. We have previously argued [7] that opening the web to provide an open DTD capability at the browser would free applications from the limitations of HTML. In this approach, any legal SGML DTD and document instance could be uploaded and delivered.

Once SGML is supported at the client, support for HyTime is technically feasible and would provide a much richer hypermedia framework.

> "What we really need (to repeat), is a general addressing mechanism similar to that offered by HyTime. If we continually reinvent addressing schemes with different syntaxes, it makes supporting all the various syntaxes a real headache. One general (and extensible) syntax is far more preferable." -- posted to the http-wg list during discussion of http 1.1 features to support addressing of media.

Video Dialtone and Video on Demand

Video Dialtone [23][11] is a framework for multimedia content and on-line consumer information services delivery via an interactive television interface. The content architecture being proposed for video dialtone networks by the DAVIC consortium is a combination of MPEG-2 (video, audio, systems, and DSM-CC) and MHEG-5 [20]. MHEG-5 is a subset of MHEG-1 [18] which, as discussed in previous work [2], has been designed for incremental delivery of interactive script-based multimedia content in a distributed environment. MHEG, unlike HyTime, is a final-form encoding and was designed specifically for real-time delivery over networks.

In comparison to MHEG, HyTime does not represent interaction and presentation aspects of multimedia content. It is not well suited for an incremental delivery model, since SGML must be parsed sequentially. Additionally, our implementation experience with both MHEG [4] and HyTime [1] indicates that an MHEG engine is significantly simpler and requires less memory, an important consideration for consumer devices.

Virtual Reality

Another WWW development, VRML (virtual reality markup language), demonstrates that it is possible to combine hyperlinking with three-dimensional scene navigation in a useful manner. Assuming that the encoding of the cyber world were done in a specially designed format such as VRML or MPEG-4, HyTime notation location addressing combined with an appropriate notation definition could be used to create a HyTime hyperlink structure over a set of virtual reality scenes or worlds.

Content-Based Retrieval

Recently there has been progress in image and video retrieval techniques suitable for multimedia databases and authoring [12]. Such algorithms are the basis for defining notations for representing anchors in visual media. As is the case for VR, HyTime notation location addressing could be used with such a notation to permit HyTime documents to link to addressed portions of visual content.

Hyperapplications

Several object-oriented application frameworks have appeared recently, and some of these support the ability to embed one application in another application's display area. These frameworks do not currently define an explicit declarative hypergraph representation that can be accessed by an arbitrary application. Rather, the linking and embedding mechanisms are accomplished programmatically. Making the hypergraph structure explicit would make it possible to support link browsers and searching and filtering facilities for links.

Persistent storage of hypergraph and the application anchors and navigation methods is needed to preserve the associative information between user sessions. HyTime provides a generic model for links and location addressing. The HyTime location addressing forms might be useful for applications which manage documents. The ultimate goal would be a hyperbase which could support both the

document architecture and the hyperapplication framework. HyTime link and location address could be a useful part of the design of such a hyperbase.

IMPLEMENTATION EXPERIENCE

A HyTime engine takes as input the parse tree produced by the SGML parser. It performs HyTime-specific validation of the parse tree, and may create special index structures to make navigation of the HyTime constructs in the document more efficient. It provides a HyTime-specific application programmer interface so that an application can transverse the document tree and locate the HyTime structures of interest. The HyTime standard does not specify the design of HyTime engines, nor does it define an API. A natural way to organize this API, as we describe in the development of the HyOctane™ HyTime engine [1], is to provide classes which are associated with each functional area such as linking, locating, scheduling, etc.

A HyTime application accesses the HyTime document structure stored in the engine and presents this to the user. User interaction may lead to presentation state change, and ultimately to navigation within the document. All presentation and interaction decisions are handled by the application. Any non-trivial HyTime application will require specific software or scriptware in order to implement the presentation and interaction semantics. This is an impediment to the development of HyTime application development and delivery. On the development side it means that an application developer faces both a DTD design task and a programming task. On the delivery side, while the HyTime document instance is portable, the application software or scriptware that interfaces with the engine is generally not. Our solution to this problem [7] is to integrate the HyTime engine with a standard script player, such as the virtual machine player specified in MHEG-3 [19]. This addresses the portability issue of the presentation/interaction software and also makes the delivery model open to arbitrary scripting languages. In this model, an application designer creates a HyTime DTD and writes a presentation script for this DTD. The presentation script is interchanged with the DTD and document instance whenever the document is to be delivered to the end user. We refer to the resulting delivery architecture as an open hyperdocument model.

The initial client-server model of our engine used a server-side OODBMS to store the parse tree directly as a tree of objects. This mapping turned out to be inefficient for storage of large numbers of documents. Subsequently we have revised the internal organization to create a compact representation of the document which is still convenient for indexing sub-structure within the document. Our measurements to date indicate that the compact OODBMS storage format that we have designed is compressed from the native text files by a ratio of 3.5:1. These measurements on based on a HyTime application we have designed called HMP (Hypermedia Presentation DTD). We have done similar measurements on storing parsed HTML 2.0 documents in the database. For small documents (1K to 3K bytes) we obtained compression ratios in the range of 1.5: 1 to 2.0:1. For a large document (about 300K bytes) we obtain a compression ratio of 3:1.

We obtain more compression for HyTime documents than HTML documents due to two factors. First, a HyTime DTD like HMP has many levels of nesting due to the structural requirements in the HyTime meta-DTD. For example, to associate a location with an image in a HMP document, the HyTime meta-DTD requires it to be three levels down from the root element of the document. This high degree of nesting results in many repetitions of an element's begin and end tags. Second, HMP documents also use a large number of attributes. Most elements in a HMP document have no data, but contain several attributes. This is due to HyTime element type forms which make extensive use of attributes. HMP is a representative HyTime DTD. Its extensive used of attributes and the high degree of element nesting are a direct result of the requirements placed on it by the HyTime meta-DTD. Our server-side preprocessing and storage technique takes advantage of these two features, leading to more efficient storage and delivery of HyTime (as well as HTML and SGML) documents compared to existing systems.

In summary, the Distributed HyOctane™ engine has three fundamental features: 1) the document model is open to any HyTime or SGML DTD, including HTML, 2) the hypermedia semantics of HyTime related to links, time/space, and anchoring are available, 3) the server stores a preparsed compact version of the document, simplifying the design of the client and leading to reduced network overhead. We have developed a HyTime application called HMP (HyperMedia Presentation). The HyOctane engine uses a OODBMS to store parsed documents as objects. It also implements the HyQ document query language. The HyOctane engine is interfaced to an http server which passes the preparsed, compact HTML and HyTime documents to the client for direct presentation. Compared to the current httpd server design, the HyOctane approach: 1) saves significant storage space at the server; 2) eliminates the need for client side parsing; 3) reduces the amount of information being transmitted over the network; 4) is the basis for incremental delivery of SGML-encoded documents; 5) facilitates server-side document structure queries.

SUMMARY

HyTime is concerned with addressing, associating, and structuring of hypermedia information. In these areas it provides a general and extensive set of architectural forms.

HyTime has attracted some interest both because of its position as an international standard and due to the elegance and innovation of its general design. On the other hand it remains enigmatic to many users due to its complexity and meta-SGML relationship. Translating multimedia documents authored in today's commercial tools to HyTime remains a complex task.

During the period of HyTime's development, relatively little had been done to deliver interactive composite multimedia content in hypermedia systems. Similarly, hypermedia query languages were also relatively novel. Consequently in these areas HyTime had a smaller reservoir of experience to draw upon.

For multimedia applications, there are significant representational limitations with regard to interactive behavior, support for scripting language integration, and presentation aspects. In these and other areas noted in the paper, the designers chose for the most part to avoid defining a semantics while at the same time defining extensive structural facilities. This strategy is in keeping with the logic markup tenet of SGML. However, for multimedia applications these omitted areas are typically an intrinsic part of an overall composition created by a designer, and must currently be expressed in an application-specific way.

ACKNOWLEDGMENTS

The author appreciates the comments of the anonymous referees who reviewed this paper, as well as comments on the final version of the paper provided by Lloyd Rutledge.

REFERENCES

1. Buford, J., Rutledge, L., Rutledge, J., and Keskin C., "HyOctane: A HyTime Engine for an MMIS", *Multimedia Systems (1)* 4, February 1994.

2. Buford, J., and Brennan, R. Multimedia Interchange. in *Multimedia Systems* (J.F. K. Buford, ed.) ACM Press/Addison-Wesley, 1994

3. Buford, J., Rutledge, L, and Rutledge J., "Integrating Object-Oriented Scripting Languages with HyTime", *Proc. IEEE Intl. Conf. on Multimedia Computing and Systems*, May 1994.

4. Buford, J.,. and Gopal. C., "MHEG and OMFI: A Comparison of Two Interchange Formats", *Proc. IEEE Intl. Conf. on Multimedia Computing and Systems*, May 1994.

5. Buford, J., Rutledge, L, and Rutledge J., "Towards Automatic Generation of HyTime Applications", *Proc. Eurographics Multimedia 94*, June 1994.

6. Buford, J., "Evaluation of a Query Language for Structured Hypermedia Documents", *Proc. DAGS 95: Electronic Publ. and the Information Superhighway*, June 1995.

7. Buford, J., "A Transfer Protocol for an Open Hyperdocument Server", *Proc. ED-MEDIA 95*, June 1995.

8. Buford, J., Rutledge, J., Rutledge L., Gopal, C., "A Distributed Open Hypermedia Server Based on the ODMG Specification", in preparation.

9. DeRose, S. and Durand. D., *HyTime: Making Hypermedia Work*, Kluwer Academic Press, 1994.

10. Erfle, R. Specification of temporal constraints in multimedia documents using HyTime. *Electronic Publishing*, vol. 6(4), Dec. 1993, pp. 397-411.

11. Furht, B., et al. Design Issues for Interactive Television Systems. *Computer (28)* 5, May 1995, pp. 25-39.

12. Gudivada, V. and V. Raghavan (eds.) Special Issue on Content-Based Retrieval. *Computer 28*(9), Sept. 1995.

13. Halasz, F., and Schwartz, M., "The Dexter Hypertext Reference Model", *CACM (37)* 2, February 1994.

14. Halasz, F. "Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems", *CACM (31)* 7, July 1988.

15. Hardman, L. van Ossergruggen, J, and Bulterman, D.C.A., "A HyTime-Compliant Interchange Format for CMIF Hypermedia Documents", in progress.

16. Hill, G., Hall, W., "Extending the Microcosm Model to a Distributed Environment", *Proc. ECHT 94*.

17. ISO/IEC IS 10744, Hypermedia/Time-based Document Structuring Language (HyTime), August 1992.

18. ISO/IEC IS 13522-1, Multimedia and Hypermedia Information Encoding (MHEG), Part 1 ASN.1 Encoding. Nov. 1995

19. ISO/IEC CD 13522-3 Multimedia and Hypermedia Information Encoding (MHEG), Part 3 Support for Scripting Languages. August 1995

20. ISO/IEC CD 13522-5 Multimedia and Hypermedia Information Encoding (MHEG), Part 5 Conformance Model, Aug 1995

21. Kappe, F., et al., Hyper-G "A New Tool for Distributed Hypermedia", *Proc. IASTED Distr. Multimedia Syst. and Appl 94*, 1994.

22. Loeffler, C., and Anderson, T. (ed.) *The Virtual Reality Casebook*. Van Nostrand Reinhold. 1995.

23. Minoli, D. *Video Dialtone Technology*. McGraw Hill, 1995.

24. Newcomb, S., "The `HyTime' Hypermedia / Time-based Document Structuring Language", *CACM (34)* 2, February 1992.

25. Rutledge, L. Buford, J., and Rutledge, J., "Applying HyTime to HTML", *Proc. IASTED Distributed Multimedia Systems and Applications 95*. August 1995.

26. Rutledge, L., Buford, J., and Rutledge, J., "Modeling Techniques for HyTime", *Proc. Multimedia Modeling 95*, November 1995.

27. Rutledge, L., Buford, J., and Rutledge, J., "Presenting HyTime Documents with HTML", submitted.